Google

Web Images Groups News Froogle Local more »

timestamp stack signed unsigned Search Advanced Search Preferences

Web Results **1** - **10** of about **25,800** for **timestamp stack signed unsigned**. (**0.23** seconds)

iAnywhere.com - Data types in embedded SQL
... 16-bit **signed** integer. short int i; **unsigned** short int i;. 32-bit **signed** integer.
... SQLDATETIME structure with fields for each part of a **timestamp**. ...
www.ianywhere.com/developer/product_manuals/ sqlanywhere/0901/en/html/ulcpen9/00000075.htm - 12k - Cached - Similar pages

ACIS Science Instrument Software Detailed Design Specification (As ...
... char **signed** 8-bit value **unsigned** char **unsigned** 8-bit value short **signed** ...
The ACIS **timestamp** counter relies on the spacecraft-supplied 1.024 MHz clock ...
acis.mit.edu/acis/sdetail-01pp/introduction.html - 19k - Cached - Similar pages

RE: [dss] Further - Request for inclussion to the requirementsdocument
... the **timestamp** and requestor identity into the XAdES structure (for example,
... of where you put the **signed/unsigned** > >attributes that you have to use, ...
lists.oasis-open.org/archives/dss/200305/msg00039.html - 11k - Cached - Similar pages

> RE: [dss] Further - Request for inclussion to the requirementsdocument
> ... profile that discusses how to incorporate the **timestamp** > and requestor identity
> ... you put the > **signed/unsigned** > > >attributes that you have to use, ...
> lists.oasis-open.org/archives/dss/200305/msg00042.html - 12k - Cached - Similar pages

Data Types and Data Typing
... Value types allocated on the **stack**; Reference types allocated on the heap ...
Sawtell) tests whether the char inherent data type is **signed** or **unsigned**. ...
www.wilsonmar.com/DataTypes.htm - 57k - Cached - Similar pages

empserver: include/misc.h Source File
... 00043 typedef **unsigned** char u_char; 00044 typedef **unsigned** short u_short; ...
updating 00074 mobility and the **timestamp** of when the game was last up. ...
www.**stack**.nl/~marcolz/empire/ doxy/include_2misc_8h-source.html - 16k - May 25, 2005 - Cached - Similar pages

[PDF] Rkit-STFIVE
File Format: PDF/Adobe Acrobat - View as HTML
... Integer Types: 8 bits : "**signed** char" and "**unsigned** char", ... locations),
trace with **timestamp**, Code Coverage and **stack**. consumption analysis. ...
www.raisonance.com/files/pdf/Rkit-ST5.PDF - Similar pages

The Old New Thing : Myth: The /3GB switch expands the user-mode ...
... it doesn't matter whether it's **signed** or **unsigned** just so that it's consitent.
... 75eb0000 75fb6000 browseui **Timestamp**: Thu Jul 08 19:25:07 2004 ...
blogs.msdn.com/oldnewthing/ archive/2004/08/12/213468.aspx - 93k - Cached - Similar pages

M88K Options
... Control how function arguments are stored in **stack** frames. ... GNU C emulates
signed integer division using the **unsigned** integer division instruction ...
docs.freebsd.org/info/gcc/gcc.info.M88K_Options.html - 9k - Cached - Similar pages

a.out(4)
... struct sys_clock file_time; /* **timestamp** */ **unsigned** int entry_space; /* index of
... All items on the **stack** are considered **signed** 32-bit integers. ...
docs.hp.com/en/B2355-90131/a.out.4.html - 162k - Cached - Similar pages

Gooooooooooogle ▶
Result Page: 1 2 3 4 5 6 7 8 9 10 **Next**

timestamp stack signed unsigned Search

Search within results | Language Tools | Search Tips | Dissatisfied? Help us improve

Google Home - Advertising Programs - Business Solutions - About Google

Terms used **timestamp stack profiling** Found **2,944** of **155,867**

Sort results by: relevance | Display results: expanded form

Save results to a Binder | Search Tips | ☐ Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Best 200 shown Relevance scale

1 Profiling Java applications using code hotswapping and dynamic call graph revelation

Mikhail Dmitriev

January 2004 **ACM SIGSOFT Software Engineering Notes , Proceedings of the fourth international workshop on Software and performance**, Volume 29 Issue 1

Full text available: pdf(1.32 MB) Additional Information: full citation, abstract, references

Instrumentation-based profiling has many advantages and one serious disadvantage: usually high performance overhead. This overhead can be substantially reduced if only a small part of the target application (for example, one that has previously been identified as a performance bottleneck) is instrumented, while the rest of the application code continues to run at full speed. The value of such a profiling technology would increase further if the code could be instrumented and de-instrumented as m ...

2 Dynamic statistical profiling of communication activity in distributed applications

Jeffrey Vetter

June 2002 **ACM SIGMETRICS Performance Evaluation Review , Proceedings of the 2002 ACM SIGMETRICS international conference on Measurement and modeling of computer systems**, Volume 30 Issue 1

Full text available: pdf(1.65 MB) Additional Information: full citation, abstract, references

Performance analysis of communication activity for a terascale application with traditional message tracing can be overwhelming in terms of overhead, perturbation, and storage. We propose a novel alternative that enables dynamic statistical profiling of an application's communication activity using message sampling. We have implemented an operational prototype, named PHOTON, and our evidence shows that this new approach can provide an accurate, low-overhead, tractable alternative for pe ...

3 Memory hierarchy: Compiler-decided dynamic memory allocation for scratch-pad based embedded systems

Sumesh Udayakumaran, Rajeev Barua

October 2003 **Proceedings of the 2003 international conference on Compilers, architecture and synthesis for embedded systems**

Full text available: pdf(213.48 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents a highly predictable, low overhead and yet dynamic, memory allocation strategy for embedded systems with scratch-pad memory. A *scratch-pad* is a fast compiler-managed SRAM memory that replaces the hardware-managed cache. It is motivated by its better real-time guarantees vs cache and by its significantly lower overheads in energy

consumption, area and overall runtime, even with a simple allocation scheme [4].Existing scratch-pad allocation methods are of two types. Firs ...

Keywords: compiler, embedded systems, memory allocation, scratch-pad

4 Quartz: a tool for tuning parallel program performance

Thomas E. Anderson, Edward D. Lazowska

April 1990 **ACM SIGMETRICS Performance Evaluation Review , Proceedings of the 1990 ACM SIGMETRICS conference on Measurement and modeling of computer systems**, Volume 18 Issue 1

Full text available: pdf(1.51 MB) Additional Information: full citation, abstract, references, citings, index terms

Initial implementations of parallel programs typically yield disappointing performance. Tuning to improve performance is thus a significant part of the parallel programming process. The effort required to tune a parallel program, and the level of performance that eventually is achieved, both depend heavily on the quality of the instrumentation that is available to the programmer. This paper describes Quartz, a new tool for tuning parallel program performance on shared memory mult ...

5 High resolution timing with low resolution clocks and microsecond resolution timer for Sun workstations

Peter B. Danzig, Stephen Melvin

January 1990 **ACM SIGOPS Operating Systems Review**, Volume 24 Issue 1

Full text available: pdf(303.49 KB) Additional Information: full citation, abstract, citings, index terms

When tuning operating system and network code, profiling programs, analyzing message interarrival times, and accurately measuring device characteristics, a high resolution clock is often indispensable, as one cannot measure service time *distributions* without one. This note describes a microsecond clock that we designed and built for Sun 3 and Sun 4 workstations[1]. One can measure average service times without a high resolution clock. This paper explains how to measure average ti ...

6 Profiling and reducing processing overheads in TCP/IP

Jonathan Kay, Joseph Pasquale

December 1996 **IEEE/ACM Transactions on Networking (TON)**, Volume 4 Issue 6

Full text available: pdf(1.21 MB) Additional Information: full citation, references, citings, index terms

7 The Jrpm system for dynamically parallelizing Java programs

Michael K. Chen, Kunle Olukotun

May 2003 **ACM SIGARCH Computer Architecture News , Proceedings of the 30th annual international symposium on Computer architecture**, Volume 31 Issue 2

Full text available: pdf(320.42 KB) Additional Information: full citation, abstract, references, citings

We describe the Java runtime parallelizing machine (Jrpm), a complete system for parallelizing sequential programs automatically. Jrpm is based on a chip multiprocessor (CMP) with thread-level speculation (TLS) support. CMPs have low sharing and communication costs relative to traditional multiprocessors, and thread-level speculation (TLS) simplifies program parallelization by allowing us to parallelize optimistically without violating correct sequential program behavior. Using a Java virtual ma ...

8 Error-free garbage collection traces: how to cheat and not get caught

Matthew Hertz, Stephen M Blackburn, J Eliot B Moss, Kathryn S. McKinley, Darko Stefanović

June 2002 **ACM SIGMETRICS Performance Evaluation Review , Proceedings of the 2002 ACM SIGMETRICS international conference on Measurement and modeling of computer systems**, Volume 30 Issue 1

Full text available: pdf(105.06 KB) Additional Information: full citation, abstract, references, citings

Programmers are writing a large and rapidly growing number of programs in object-oriented languages such as Java that require garbage collection (GC). To explore the design and evaluation of GC algorithms quickly, researchers are using simulation based on traces of object allocation and lifetime behavior. The *brute force* method generates perfect traces using a whole-heap GC at every potential GC point in the program. Because this process is prohibitively expensive, researchers often use < ...

9 Object equality profiling

Darko Marinov, Robert O'Callahan

October 2003 **ACM SIGPLAN Notices , Proceedings of the 18th annual ACM SIGPLAN conference on Object-oriented programing, systems, languages, and applications**, Volume 38 Issue 11

Full text available: pdf(577.47 KB) Additional Information: full citation, abstract, references, citings, index terms

We present *Object Equality Profiling* (OEP), a new technique for helping programmers discover optimization opportunities in programs. OEP discovers opportunities for replacing a set of equivalent object instances with a single representative object. Such a set represents an opportunity for automatically or manually applying optimizations such as hash consing, heap compression, lazy allocation, object caching, invariant hoisting, and more. To evaluate OEP, we implemented a tool to help prog ...

Keywords: Java language, object equality, object mergeability, profile-guided optimization, profiling, space savings

10 Measuring limits of parallelism and characterizing its vulnerability to resource constraints

Lawrence Rauchwerger, Pradeep K. Dubey, Ravi Nair

December 1993 **Proceedings of the 26th annual international symposium on Microarchitecture**

Full text available: pdf(1.39 MB) Additional Information: full citation, references

11 Experience with "link-up notification" over a mobile satellite link

Martin Duke, Thomas R. Henderson, Jeff Meegan

July 2004 **ACM SIGCOMM Computer Communication Review**, Volume 34 Issue 3

Full text available: pdf(429.96 KB) Additional Information: full citation, abstract, references

Over paths characterized by extended outage periods, a TCP connection can suffer a severe performance penalty due to its Retransmission Timeout (RTO) backoff mechanism. If outages are long enough, the RTO can grow large enough to cause unacceptably long pauses when the link is eventually restored. One proposed solution is "Link-Up Notification" (LUN), which involves an intermediate device that can detect the link state. When the link is restored, the device immediately sends a packet that cau ...

12 Integrating pointer variables into one-way constraint models

Brad Vander Zanden, Brad A. Myers, Dario A. Giuse, Pedro Szekely

June 1994 **ACM Transactions on Computer-Human Interaction (TOCHI)**, Volume 1 Issue 2

Full text available: pdf(3.71 MB) Additional Information: full citation, abstract, references, citings, index

terms

Pointer variables have long been considered useful for constructing and manipulating data structures in traditional programming languages. This article discusses how pointer variables can be integrated into one-way constraint models and indicates how these constraints can be usefully employed in user interfaces. Pointer variables allow constraints to model a wide array of dynamic application behavior, simplify the implementation of structured objects and demonstrational systems, and improve ...

Keywords: Garnet, constraints, development tools, incremental algorithms

13 vic: a flexible framework for packet video

Steven McCanne, Van Jacobson

January 1995 **Proceedings of the third ACM international conference on Multimedia**

Full text available: html(67.64 KB) Additional Information: full citation, references, citings, index terms

Keywords: conferencing protocols, digital video, image and video compression and processing, multicasting, networking and communication

14 Optimizing 10-Gigabit Ethernet for Networks of Workstations, Clusters, and Grids: A Case Study

Wu-chun Feng, Justin (Gus) Hurwitz, Harvey Newman, Sylvain Ravot, R. Les Cottrell, Olivier Martin, Fabrizio Coccetti, Cheng Jin, Xiaoliang (David) Wei, Steven Low

November 2003 **Proceedings of the 2003 ACM/IEEE conference on Supercomputing**

Full text available: pdf(209.19 KB) Additional Information: full citation, abstract

This paper presents a case study of the 10-Gigabit Ethernet (10GbE) adapter from Intel R . Specifically, with appropriate optimizations to the configurations of the 10GbE adapter and TCP, we demonstrate that the 10GbE adapter can perform well in local-area, storage-area, system-area, and wide-area networks. For local-area, storage-area, and system-area networks in support of networks of workstations, network-attached storage, and clusters, respectively, we can achieve over 7-Gb/s end-to-end thro ...

15 Reliability and security: Memory overflow protection for embedded systems using run-time checks, reuse and compression

Surupa Biswas, Matthew Simpson, Rajeev Barua

September 2004 **Proceedings of the 2004 international conference on Compilers, architecture, and synthesis for embedded systems**

Full text available: pdf(253.51 KB) Additional Information: full citation, abstract, references, index terms

Out-of-memory errors are a serious source of unreliability in most embedded systems. Applications run out of main memory because of the frequent difficulty of estimating the memory requirement before deployment, either because it depends on input data, or because certain language features prevent estimation. The typical lack of disks and virtual memory in embedded systems has two serious consequences when an out-of-memory error occurs. First, there is no swap space for the application to grow in ...

Keywords: data compression, heap overflow, out-of-memory errors, reliability, reuse, runtime checks, stack overflow

16

Profile-directed optimization of event-based programs

Mohan Rajagopalan, Saumya K. Debray, Matti A. Hiltunen, Richard D. Schlichting
May 2002 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2002 Conference on Programming language design and implementation**, Volume 37 Issue 5

Full text available: pdf(167.69 KB) Additional Information: full citation, abstract, references, citings, index terms

Events are used as a fundamental abstraction in programs ranging from graphical user interfaces (GUIs) to systems for building customized network protocols. While providing a flexible structuring and execution paradigm, events have the potentially serious drawback of extra execution overhead due to the indirection between modules that raise events and those that handle them. This paper describes an approach to addressing this issue using static optimization techniques. This approach, which explo ...

Keywords: events, handlers, profiling

17 Performance monitoring: TEST: a tracer for extracting speculative threads

Michael Chen, Kunle Olukotun
March 2003 **Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization**

Full text available: pdf(1.76 MB) Publisher Site Additional Information: full citation, abstract, references, citings, index terms

Thread-level speculation (TLS) allows sequential programs to be arbitrarily decomposed into threads that can be safely executed in parallel. A key challenge for TLS processors is choosing thread decompositions that speedup the program. Current techniques for identifying decompositions have practical limitations in real systems. Traditional parallelizing compilers do not work effectively on most integer programs, and software profiling slows down program execution too much for real-time analysis. ...

18 Frequent value locality and its applications

Jun Yang, Rajiv Gupta
November 2002 **ACM Transactions on Embedded Computing Systems (TECS)**, Volume 1 Issue 1

Full text available: pdf(1.28 MB) Additional Information: full citation, abstract, references, citings, index terms

By analyzing the behavior of a set of benchmarks, we demonstrate that a small number of distinct values tend to occur very frequently in memory. On an average, only eight of these frequent values were found to occupy 48% of memory locations for the benchmarks studied. In addition, we demonstrate that the identity of frequent values remains stable over the entire execution of the program and these values are scattered fairly uniformly across the allocated memory. We present three different ...

Keywords: Frequently occurring values, encoding techniques, low power data bus, low power data cache, value profiling

19 LLVA: A Low-level Virtual Instruction Set Architecture

Vikram Adve, Chris Lattner, Michael Brukman, Anand Shukla, Brian Gaeke
December 2003 **Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture**

Full text available: pdf(196.08 KB) Additional Information: full citation, abstract, index terms

A virtual instruction set architecture (V-ISA) implementedvia a processor-specific software translation layercan provide great flexibility to processor designers. Recentexamples such as Crusoe and DAISY, however, haveused existing hardware instruction sets as virtual ISAs,which complicates translation and optimization. In fact,there has been little research

on specific designs for a virtualISA for processors. This paper proposes a novel virtualISA (LLVA) and a translation strategy for implementi ...

20 Memory Profiling using Hardware Counters

Marty Itzkowitz, Brian J. N. Wylie, Christopher Aoki, Nicolai Kosche
November 2003 **Proceedings of the 2003 ACM/IEEE conference on Supercomputing**

Full text available: pdf(117.74 KB) Additional Information: full citation, abstract

Although memory performance is often a limiting factor in application performance, most tools only show performance data relating to the instructions in the program, not to its data. In this paper, we describe a technique for directly measuring the memory profile of an application. We describe the tools and their user model, and then discuss a particular code, the MCFbenchmark from SPEC CPU 2000. We show performance data for the data structures and elements, and discuss the use of the data to im ...

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Results for "(stack unsigned<in>metadata)"
Your search matched **0** of **1164322** documents.
A maximum of **100** results are displayed, **25** to a page, sorted by **Relevance** in **Descending** order.

» View Session History

» New Search

Modify Search

(stack unsigned<in>metadata)

Check to search only within this results set

Display Format: Citation Citation & Abstract

» Key	
IEEE JNL	IEEE Journal or Magazine
IEE JNL	IEE Journal or Magazine
IEEE CNF	IEEE Conference Proceeding
IEE CNF	IEE Conference Proceeding
IEEE STD	IEEE Standard

No results were found.

Please edit your search criteria and try again. Refer to the Help pages if you need assistance revisir

Results for "(probe timestamp<in>metadata)"

e-mail

Your search matched **0** of **1164322** documents.

A maximum of **100** results are displayed, **25** to a page, sorted by **Relevance** in **Descending** order.

» View Session History

» New Search

Modify Search

(probe timestamp<in>metadata)

Check to search only within this results set

Display Format: Citation Citation & Abstract

» Key	
IEEE JNL	IEEE Journal or Magazine
IEE JNL	IEE Journal or Magazine
IEEE CNF	IEEE Conference Proceeding
IEE CNF	IEE Conference Proceeding
IEEE STD	IEEE Standard

No results were found.

Please edit your search criteria and try again. Refer to the Help pages if you need assistance revisir